

## **REMARKS**

### **Introduction**

Claims 2, 6 and 10 have been canceled. Claims 1, 3-5, and 7-9 have been amended. Claims 1, 3-5, and 7-9 remain in the application, of which claim 1 is in independent form.

### **Objections to the Specification**

The specification was objected to because the trademarked term "JAVA" was not presented in uppercase characters throughout the specification. By this Amendment, the Abstract and specification have been amended to recite "JAVA" in uppercase characters, and thus applicant believes the objections to be obviated, and withdrawal of the objections is requested.

### **Rejections under 35 U.S.C. § 112**

Claims 1-10 stand rejected under 35 U.S.C. § 112, second paragraph, for allegedly being indefinite due to certain informalities. Applicants have amended the claims to attend to the cited informalities. Accordingly, applicants respectfully submit that amended claims 1, 3-5, and 7-9, are not indefinite, and withdrawal of the rejections to claims 1, 3-5, and 7-9 under 35 U.S.C. § 112, second paragraph, is respectfully requested.

### **Rejections under 35 U.S.C. § 101**

Claims 1-10 stand rejected based on 35 U.S.C. § 101 for allegedly being directed to non-statutory subject matter.

The first part of the rejection contends that the claimed method is directed to a computer program *per se* and thus is non-statutory. Applicants submit that while "computer programs claimed as computer listings ... are neither computer components nor statutory

processes," and are thus "descriptive material *per se* and hence nonstatutory," (*MPEP* § 2106.01(I)), applicants' claimed invention is not a computer listing, but a process.

Claims remain statutory "when a computer program is used in a computerized process where the computer executes instructions set forth in the computer program." *Id.*

The second part of the rejection contends that the claimed method does not produce a useful, concrete and tangible result. Applicants have amended claim 1 to recite "displaying the human-readable form to a user." Thus, the claimed method displays human readable results of a difference between two objects. This result is useful (a user can easily determine differences between objects), concrete (the results are repeatable) and tangible (real world, useful results are produced). (*See MPEP* § 2106(IV)(C)(2)(2)(a)-(c)).

With regard to the Examiner's comments regarding optional recitations, applicants have amended claim 1 to attend to the Examiner's concerns regarding optional recitations.

Accordingly, applicants respectfully submit that claims 1, 3-5, and 7-9, are directed to statutory subject matter, and withdrawal of the rejections to claims 1, 3-5, and 7-9 under 35 U.S.C. § 101 is respectfully requested.

**Rejections under 35 U.S.C. § 103(a)**

Claims 1-5, 9 and 10 stand rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,826,716 (*Mason*) in view of "Java Performance Tuning," Section 7.5, "Recursion and Stacks," September 2000 (*Shirazi*).

Amended independent claim 1 of the present application is directed to a computer implemented method for "comparing a first object and a second object in an object-oriented operating system." The method can include, *inter alia*, "determining whether the first object is equal to the second object; and if said objects are not equal," the method includes the steps of "obtaining one or more methods from said first object and said second object," "determining

whether the one or more methods from said first object are equal to the one or more methods from said second object."

The method also includes "generating a document comprising a listing of differences between the methods," "transforming the document into a human-readable form," and "displaying the human-readable form to a user."

Thus, as described in the specification of the present application, to solve the problem of there presently being no universally defined method to determine the differences between two objects, the claimed method beneficially provides for a utility for comparing two objects in an object-oriented operating system and recording the differences so that they may be put in human-readable form.

*Mason* describes a system for generating "test programs" for testing Java Web applications. (*Mason* at Abstract). The generated test programs are designed to test "more combinations of actions, web application configurations or scenarios." (*Id.* at col. 2, lns. 55-57). *Mason* describes identifying modules, and identifying a "QOS [Quality of Service] element." (*Id.* at col. 2, lns. 66-67 to col. 3, lns. 1-2). Also described, are identifying a "user identification and a user password ... and testing the software resource to be tested by use of the Java test code, including passing as parameters to the Java test code at run time the user identification and user password." (*Id.* at col. 3, lns. 3-9).

*Mason* further describes extracting data from a deployment descriptor to facilitate generation of the Java test code. (*Id.* at col. 12, lns. 23-25). "In typical embodiments, testing the software resource further includes creating ...an instance of the JavaBean (450), invoking (448), in the created instance, (450), the software resource to be tested ... and reporting (444) whether invoking the protected JavaBean method succeeded." (*Id.* at col. 12, lns. 40-46 and FIG. 4)(emphasis added). The test code described by *Mason* can generate different kinds of tests,

"each targeted at a specific aspect of quality or verification. For instance, correctness, load/stress test, reliability, long running tests, and so on." (*Id.* at col. 15, lns. 32-35). Thus, *Mason* describes a system for generating a test program for testing a Java Web application to determine if it succeeds or fails.

Applicants submit that *Mason* does not teach, suggest or provide motivation for all of the features recited by amended claim 1 of the present application. While claim 1 of the present application claims a method for "comparing a first object and a second object in an object-oriented operating system" and "generating a document comprising a listing of differences between the methods," in stark contrast, *Mason* describes a system for testing various parameters of Java Web applications to determine if the applications succeed. Thus, the system described by *Mason* is very different from the method claimed by claim 1 of the present application.

Moreover, *Mason* does not describe other features recited by claim 1. For example, *Mason* does not teach, suggest or provide motivation for "determining whether the first object is equal to the second object; and if said objects are not equal," or "obtaining one or more methods from said first object and said second object." In addition, *Mason* does not teach, suggest or provide motivation for the recited "determining whether the one or more methods from said first object are equal to the one or more methods from said second object," "generating a document comprising a listing of differences between the methods," "transforming the document into a human-readable form," or "displaying the human-readable form to a user."

*Shirazi* describes the use of programming techniques to convert recursive method calls to iterative method calls. (*Shirazi* at page 1). *Shirazi* does not make up for the shortcomings of *Mason*. For example, *Shirazi*, either alone, or in combination with *Mason*, does not teach, suggest, or provide motivation for "comparing a first object and a second object in an object-oriented operating system." Nor does any *Mason-Shirazi* combination teach, suggest or provide

motivation for "determining whether the first object is equal to the second object; and if said objects are not equal," "obtaining one or more methods from said first object and said second object," and "determining whether the one or more methods from said first object are equal to the one or more methods from said second object." Still further, the *Mason-Shirazi* combination does not teach, suggest or provide motivation for "generating a document comprising a listing of differences between the methods," "transforming the document into a human-readable form." and "displaying the human-readable form to a user."

Accordingly, for at least these reasons, claim 1 is deemed to distinguish patentably over any hypothetical *Mason-Shirazi* combination.

Claims 3-5, and 9 depend from, and further narrow and define, claim 1, that has been discussed above and is believed to be allowable over any *Mason-Shirazi* combination. Accordingly, for at least these reasons, claims 3-5, and 9 are deemed to distinguish patentably over any hypothetical *Mason-Shirazi* combination.

Claims 6-8 stand rejected under 35 U.S.C. 103(a) as being unpatentable over *Mason* in view of U. S. Patent No. 6,662,312 (*Keller*).

Claims 6-8 depend from, and further narrow and define, claim 1, that has been discussed above and is believed to be allowable over *Mason*.

*Keller* is directed to a software testing automation system, but does not make up for the deficiencies of *Mason*. Accordingly, for at least these reasons, claims 6-8 are deemed to distinguish patentably over any hypothetical *Mason-Keller* combination.


Thus, applicants submit that each of the claims of the present application are patentable over each of the references of record, either taken alone, or in any proposed hypothetical combination. Accordingly, withdrawal of the rejections to the claims is respectfully requested.

**Conclusion**

In view of the above remarks, reconsideration and allowance of the present application is respectfully requested.

Respectfully submitted,

Date: 5 December 2016

  
\_\_\_\_\_  
By: James Dobrow  
Attorney for Applicant  
Registration No. 46,666

Mail all correspondence to:

Docket Administrator  
Lowenstein Sandler PC  
65 Livingston Avenue  
Roseland, NJ 07068